String Pattern Recognition Using Evolving Spiking Neural Networks and Quantum Inspired Particle Swarm Optimization

Haza Nuzly Abdull Hamed[1], Nikola Kasabov[1], Zbynek Michlovský[2], and Siti Mariyam Shamsuddin[3]

[1] Knowledge Engineering and Discovery Research Institute (KEDRI), Auckland University of Technology, New Zealand
{hnuzly,nkasabov}@aut.ac.nz
[2] Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic
imichlov@fit.vutbr.cz
[3] Soft Computing Research Group, Universiti Teknologi Malaysia, Malaysia
mariyam@utm.my

Abstract. This paper proposes a novel method for string pattern recognition using an Evolving Spiking Neural Network (ESNN) with Quantum-inspired Particle Swarm Optimization (QiPSO). This study reveals an interesting concept of QiPSO by representing information as binary structures. The mechanism optimizes the ESNN parameters and relevant features using the wrapper approach simultaneously. The N-gram kernel is used to map Reuters string datasets into high dimensional feature matrix which acts as an input to the proposed method. The results show promising string classification results as well as satisfactory QiPSO performance in obtaining the best combination of ESNN parameters and in identifying the most relevant features.

Keywords: String Kernels, Text Classification, Evolving Spiking Neural Network, Particle Swarm, Quantum Computing.

1 Introduction

String pattern recognition is an approach for determining which group a string belongs to, according to its contents. This task, despite being quite challenging, is very important to certain areas such as internet security and virus detection. Strings can be texts, musical symbols or others which are not necessarily in numerical formats. Since most classifier algorithms can only accept numerical values, transformation from strings to numerical values is required. String kernels are a well-known method to transform string input values into high dimensional input vectors. There are several well-known string kernels such as Bag of Words (BOW) and N-gram Kernels. Output from the kernel process which is the kernel matrix is used as an input to the algorithm for classification, clustering or ranking tasks. This technique is quite simple yet quite effective to transform input from string to the numerical value.

The third generation neural network simulates how biological neurons send and receive information based on spikes. This neural simulation increases the level of realism where the neuron is only fired when it reaches a specific value of electrical charge. Several improvements and variants of the spiking neuron model have been developed. According to Kasabov [1], the Evolving Spiking Neural Network (ESNN) as proposed by Wysoski *et al.* [2] states that the output neuron evolves based on weight vector similarities. In this study, the ESNN model is selected as the string classifier. Like other neural network models, the correct combination of ESNN parameters could influence the classification performance of the network. Apart from parameter value, the more features used, does not necessarily result in better or higher levels of classification accuracy. In some cases, having fewer significant features might help the network to reduce the processing time and produce good classifications. Therefore, feature optimization could be considered as an important pre-processing tool in a classification task. Due to this reason, Quantum-inspired Particle Swarm Optimization (QiPSO) is proposed in this study as the new optimizer model to investigate its efficiency in optimizing ESNN parameters as well as selecting the most relevant features from the string kernel matrix which both have a direct influence to the classification accuracy.

This paper is organized as followed; Section 2 briefly explains the ESNN model and Section 3 discusses the chosen optimizer model; the QiPSO and how it is derived from standard PSO. Section 4 deals with the introduction to string kernels. The framework of the proposed method is presented in Section 5. This section explains how the features are selected from the string matrix and ESNN parameters are optimized using QiPSO. Section 6 provides the details of the experiment setup and results, followed by conclusion of this study with details on future work in Section 7.

2 Evolving Spiking Neural Network

Hopfield [3] presented a model of spiking neurons in 1995. Since then, there have been several enhancements and variants of the spiking neuron model. The architecture of ESNN in this paper is based on Wysoski's model [2]. This model consists of the encoding method of real value data to spike time, network model and learning method.

There are several information encoding methods in Spiking Neural Network (SNN). However, our implementation is based on Population Encoding as proposed in [4] where a single input value is encoded to multiple input neurons of M. Each input neuron represents a certain spikes of firing time. The firing time of a input neuron i is calculated using the intersection of Gaussian function. The centre is calculated using Equation 1 and the width is computed using Equation 2 with the variable interval of $[I_{\min}, I_{\max}]$. The parameter β controls the width of each Gaussian receptive field.

$$\mu = I_{\min} + (2*i-3)/2*(I_{\max} - I_{\min})/(M-2)\,. \tag{1}$$

$$\sigma = 1/\beta(I_{\max} - I_{\min})/(M-2) \quad where \quad 1 \le \beta \le 2\,. \tag{2}$$

Thorpe [5] proposed a simplified spiking neuron model referred to as the Fast Integrate and Fire Model. The fundamental aspect of this model is that the earliest spikes received by a neuron will have a stronger weight compared to the later spikes. Once the neuron reaches a certain amount of spikes and the Post-Synaptic Potential (PSP) exceeds the threshold value, it will fire and becomes disabled. The neuron in this model can only fire once. The computation of PSP of neuron *i* is presented in Equation 3,

$$u_i = \begin{cases} 0 & if \quad fired \\ \sum w_{ji} * Mod_i^{order(j)} & else \end{cases} \quad (3)$$

where w_{ji} is the weight of pre-synaptic neuron *j*; Mod_i is a parameter called modulation factor with interval of [0,1] and *order(j)* represents the rank of spike emitted by the neuron. The *order(j)* starts with 0 if it spikes first among all pre-synaptic neuron and increases according to the firing time.

In the One-pass Learning algorithm, each training sample creates a new output neuron. Trained threshold value and the weight pattern for the particular sample are stored in the neuron repository. However, if the weight pattern of the trained neuron is considered too similar with the neuron in the repository, the neuron will merge into the most similar one. The merging process involves modifying the weight pattern and the threshold to the average value. Otherwise, it will be added as a newly trained neuron to the repository. The major advantage of ESNN is the ability of the trained network to produce new samples without retraining. The detailed ESNN learning algorithm as described in [6] is depicted below. More details on ESNN can be found in [2], [6] and [7].

Algorithm 1. ESNN Training Algorithm

```
1:  Encode input sample into firing time of pre-synaptic neuron $j$
2:  Initialize neuron repository $R = \{ \}$
3:  Set ESNN parameters $Mod = [0,1]$, $C = [0,1]$ and $Sim = [0,1]$
4:  for all input sample $i$ belong to the same output class do
5:      Set weight for all pre-synaptic neuron where: $w_j = (Mod)^{order(j)}$
6:      Calculate $PSP_{\max(i)} = \sum w_j * Mod^{order(j)}$
7:      Get PSP threshold value $\theta_i = PSP_{\max(i)} * C$
8:      if the trained weight vector <= $Sim$ of trained weight in $R$ then
9:          Merge weight and threshold value with most similar neuron
10:         $w = \frac{w_{new} + w * N}{N+1}$
11:         $\theta = \frac{\theta_{new} + \theta * N}{N+1}$
12:         where $N$ is number of merge before
13:     else
14:         Add new neuron to output neuron repository $R$
15:     end if
16: end for (Repeat to all input samples for other output class)
```

3 Quantum Inspired Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population based optimization technique developed by Eberhart and Kennedy in 1995 [8]. Individuals in PSO work together to solve a given problem by responding to their own performance and the performance of the other particles in the swarm. Each particle has their own fitness value calculated during the optimization process and the best fitness value achieved so far is stored and normally referred to as personal best or individual best (*pbest*). The overall best fitness value obtained by any particle in the population so far is called global best (*gbest*) which stores the solution. Every particle is accelerated towards a new position by calculating the position velocity where the value of *pbest* and *gbest* would influence the direction of the particle in the next iteration. Equation 4 illustrates the velocity update and Equation 5 is the calculation of the new particle position.

$$v_n = w * v_{n_{t-1}} + c_1 * rand() * (g_{best_n} - x_n) + c_2 * rand() * (p_{best_n} - x_n) . \tag{4}$$

$$x_n = x_{n_{t-1}} + v_n . \tag{5}$$

Value of the random number is between 0 and 1. C_1 and C_2 control the particle acceleration towards personal best or global best.

Han and Kim proposed a Quantum Evolutionary Algorithm (QEA) in 2002 [9] which was inspired by the concept of quantum computing. According to the classical computing concept, information is represented in bits where each bit must hold either 0 or 1. However, in quantum computing, information is instead represented by a qubit in which a value of a single qubit could be 0, 1, or a superposition of both. Superposition allows the possible states to represent both 0 and 1 simultaneously based on its probability. The quantum state is modeled by the Hilbert functions and is defined as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ where α and β are complex numbers defining probabilities at which the corresponding state is likely to appear (when a qubit collapses, for instance, when reading or measuring). Probability fundamentals require that $|\alpha|^2 + |\beta|^2 = 1$, where $|\alpha|^2$ gives the probability that a qubit is in the OFF (0) state and $|\beta|^2$ gives the probability that a qubit is in the ON (1) state. General notation for an individual with several qubits can be defined as $\begin{bmatrix} \alpha_1 & | & \alpha_2 & | & \cdots\cdots & | & \alpha_m \\ \beta_1 & | & \beta_2 & | & \cdots\cdots & | & \beta_m \end{bmatrix}$.

QiPSO was first discussed in [10] and there are several variants of QiPSO. The main idea of QiPSO is to use a standard PSO function to update the particle position represented in a qubit. In order for PSO to update the probability of a qubit, the quantum angle, θ, is used. Quantum angle θ can be represented as $\begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}$, and it is equivalent to $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ which satisfies the probability fundamental of $|\sin(\theta)|^2 + |\cos(\theta)|^2 = 1$. Therefore, the qubits value can be replaced by $[\theta_1 \mid \theta_2 | \ldots\ldots | \theta_m]$ in QiPSO. The velocity update formula in standard PSO is modified to get a new quantum angle which is translated to the new probability of the qubit by using Equation 6.

$$\Delta\theta_n = w * \Delta\theta_{n_{t-1}} + c_1 * rand() * (\theta_{gbest_n} - \theta_n) + c_2 * rand() * (\theta_{pbest_n} - \theta_n) \,. \tag{6}$$

Then, based on the new θ velocity, the new probability of α and β is calculated using a rotation gate as follows:

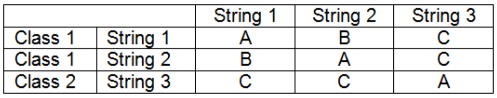
$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) \\ \sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \begin{bmatrix} \alpha_{t-1} \\ \beta_{t-1} \end{bmatrix} . \tag{7}$$

or simply by replacing the rotation gate with $\theta = \theta_{t-1} + \Delta\theta$ where θ is the new quantum angle of the quantum particle position.

4 String Kernels

In order to allow our proposed method to operate on string data, string kernels were used to transform the input data to the desired input format. Kernel provides the classifier algorithm with the capability of mapping the original non-linear separable data into a higher-dimensional space which is linearly separable. This method is normally referred as the Kernel Trick [11].

The *n*-grams approach has been chosen where *n*-grams are *n* adjacent characters (substring) from the alphabet *A* [12]. For example, if *n*=3 for the string "KEDRI", the trigrams output will be {KED, EDR, DRI}. Based on this approach, the similarity between strings is calculated and comes out with the kernel matrix. This process is illustrated in Fig. 1.



		String 1	String 2	String 3
Class 1	String 1	A	B	C
Class 1	String 2	B	A	C
Class 2	String 3	C	C	A

Fig. 1. Kernel matrix from three strings. String 1 and 2 are class 1 and String 3 is class 2. Comparison between same strings will give the highest similarity value of A, the similarity calculation between the same classes will give value of B which is slightly lower than A while the similarity value between different classes is C which is the lowest. Based on this similarity value, the feature pattern between input samples can be produced.

5 A Proposed ESNN-QiPSO for String Pattern Recognition

ESNN is chosen as the classifier of the string data in this experiment. String data is translated into the input features using *n*-gram kernel. In order to produce a better classification accuracy, all three ESNN parameters namely Modulation Factor (*Mod*), Proportion Factor (*C*) and Neuron Similarity (*Sim*) as well as input features are optimized using QiPSO. Optimization of the ESNN using QEA was addressed by Schliebs *et al.* [6] recently. However, this paper introduces QiPSO as a new optimizer for ESNN. From the well known wrapper approach, QiPSO interacts with an induction method, and in this case, ESNN; produces the best set of the ESNN parameters and identifies relevant features. All particles are initialized with a random set of binary values and from there they interact with each other based on classification accuracy. Since there are two components to be optimized, each particle is divided into two parts. The first part of each particle holds the feature mask value. This value is used for the feature optimization while the other part holds a binary string for parameter optimization.

For the parameter optimization, a set of qubits represents the parameter values. Since information held by a particle is in a binary representation, a conversion into a real value is required. For this task, the Gray code method is chosen since it is proven to be a simple and effective way to represent real values with a binary representation. In feature optimization, value 1 indicates the feature is selected, and value 0, otherwise. Input features are taken from the kernel matrix. Fig. 2 illustrates the proposed design of ESNN-QiPSO for string classification.

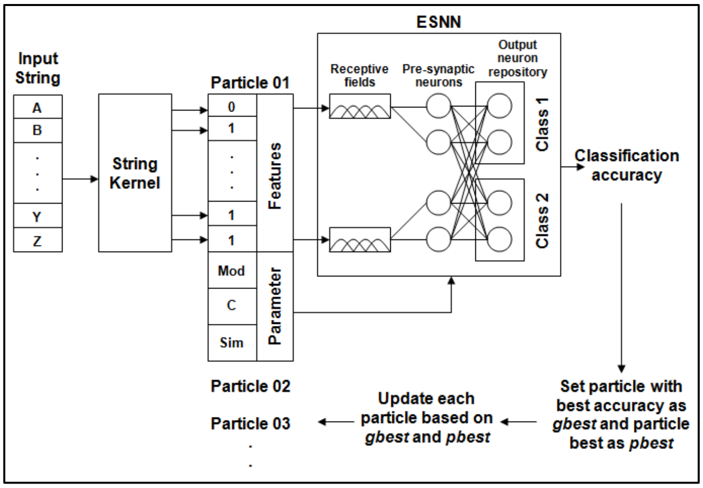


Fig. 2. A proposed ESNN-QiPSO for string classification

6 Experimental Setup and Results

We have used the labeled and reformatted Reuters 21578 string dataset from UCI Repository [13]. Only relevant information from tags topic, title and body text were extracted and some unknown tags such as "&", "$" were removed. Finally, all characters were changed to lowercase. The problem consisted of 150 samples with 38 from acquisition, 36 from corn, 38 from crude and 38 from earn. Parameters chosen were n=4 and $\lambda = 0.5$ in n-gram kernels where $\lambda = [0, 1]$ represents the number of occurrences. 3-fold cross validation was used in this experiment.

Receptive fields were used to produce a weight pattern or weight vector of a particular sample that can identify the output class. Different numbers of receptive fields for each dataset influenced the accuracy of the result. From our preliminary experiment, 10 receptive fields were chosen. 20 particles have been used to explore the solution.

Number of dimension refers to the number of variables to be optimized by QiPSO which are three ESNN parameters and 150 features. Since all three parameters ranged between 0 and 1, six qubits were sufficient to represent the real value. C_1 and C_2 were set to 0.05 which meant a balanced exploration between *gbest* and *pbest* as well as the inertia weight $w = 1.5$. The dataset has been applied to ESNN with feature optimization and also ESNN with all features, both algorithms with parameter optimization. Because of the computation complexity, we run the experiment for six continuous times and the average result was computed in 300 iterations for both algorithms.

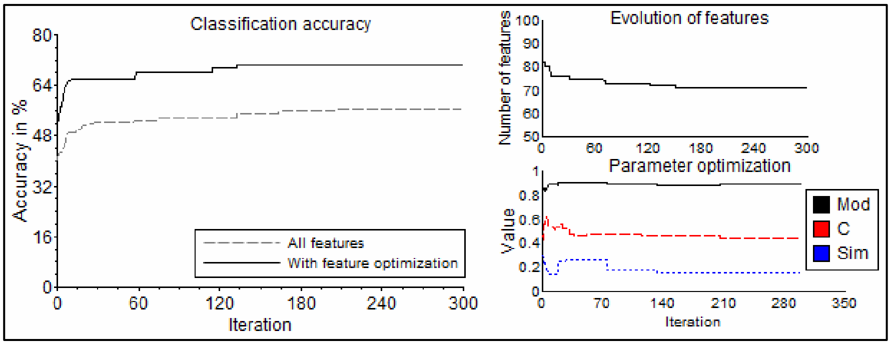


Fig. 3. Average classification result and parameter optimization

Fig. 3 shows the evolution of classification accuracy. The average accuracy for the ESNN with feature optimization is above 70% compared to ESNN using all features with the average of 55%. The poor accuracy of ESNN with all features is due to several input features from the kernel matrix containing information which cannot be used to differentiate output classes. These irrelevant features act as a noise which leads to a low classification accuracy. However, ESNN with feature optimization is able to reduce these irrelevant features; hence higher classification accuracy is obtained. From the total number of 150 input features, QiPSO is capable of reducing the features up to 70 features in 300 iterations. We strongly believe that these 70 features are the significant features because of its capability to produce higher accuracy. Since the 20 particles started the evaluation process by picking random features, we found that the average number of initial features selected by the *gbest* particle is around 80 features. The *gbest* particle always keeps best information and the best accuracy. Other particles update their position according to *gbest* as well as *pbest* until the new best particle is met. This procedure is repeated to eliminate irrelevant features for better identification of the most relevant features.

The similar procedure was also used to find the best combination of ESNN parameters. In this study, QiPSO manages to optimize binary string information which represents the ESNN parameter values. These parameters are in pair and are very closely related to each other. Overall, all three parameters evolve steadily towards a certain optimal value, and with correct combination can lead to better classification accuracy.

7 Conclusion and Future Work

This paper presents a novel method for string classification using ESNN-QiPSO. The results have shown that ESNN with parameter optimization and with using a small number of features produces promising results that is significant for future exploration. We have done some preliminary investigation to feed the string kernel matrix into Backpropagation Multi Layer Perceptron. Given the learning rate is set to 0.1, momentum rate is 0.9 with 120 hidden neurons, the training classification accuracy is around 55% in 300 iterations with testing result is around 50%. Since this experiment is in a very early stage and more experiments need to be conducted, we are planning to publish the result in the future. For future work, we are also going to apply our proposed method to real data from National Institute of Information and Communications Technology (NiCT), Japan. Other work includes how to find a more effective method for choosing the most relevant features and eliminating irrelevant features. Recurrent ESNN and integrative Probabilistic Spiking Neural Network as proposed in [14] will also be explored for string classification.

Acknowledgments

We are grateful to the National Institute of Information and Communications Technology (NiCT) Tokyo, Japan for providing us the data, problem specification and support.

References

1. Kasabov, N.: Evolving Connectionist Systems: The System Engineering Approach, 2nd edn. Springer, New York (2007)
2. Wysoski, S.G., Benuskova, L., Kasabov, N.: On-Line Learning with Structural Adaptation in a Network of Spiking Neurons for Visual Pattern Recognition. In: Kollias, S.D., Stafylopatis, A., Duch, W., Oja, E. (eds.) ICANN 2006. LNCS, vol. 4131, pp. 61–70. Springer, Heidelberg (2006)
3. Hopfield, J.: Pattern Recognition Computation Using Action Potential Timing for Stimulus Representation. Nature 376, 33–36 (1995)
4. Bohte, S.M., Kok, J.N., La Poutre, H.: Error-Backpropagation in Temporally Encoded Networks of Spiking Neurons. Neurocomputing 48(1) (2002)
5. Thorpe, S.J.: How Can The Human Visual System Process A Natural Scene in Under 150ms? Experiments and Neural Network Models. In: ESANN (1997)
6. Schliebs, S., Defoin-Platel, M., Kasabov, N.: Integrated Feature and Parameter Optimization for an Evolving Spiking Neural Network. In: Köppen, M., et al. (eds.) ICONIP 2008, Part I. LNCS, vol. 5506, pp. 1229–1236. Springer, Heidelberg (2009)
7. Schliebs, S., Defoin-Platel, M., Worner, S., Kasabov, N.: Integrated Feature and Parameter Optimization for an Evolving Spiking Neural Network: Exploring Heterogeneous Probabilistic Models. Neural Networks 22, 623–632 (2009)
8. Eberhart, R., Kennedy, J.: A New Optimizer Using Particle Swarm Theory. In: Proc. Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp. 39–43. IEEE Press, NJ (1995)

9. Han, K.H., Kim, J.H.: Quantum-Inspired Evolutionary Algorithm for a Class of Combinatorial Optimization. IEEE Transactions on Evolutionary Computation 6, 580–593 (2002)
10. Sun, J., Feng, B., Xu, W.: Particle Swarm Optimization with Particles Having Quantum Behavior. In: Proc. Cong. Evolutionary Computation, CEC 2004, vol. 1, pp. 325–331 (2004)
11. Aizerman, M., Braverman, E., Rozonoer, L.: Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning. Automation and Remote Control 25, 821–837 (1964)
12. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text Classification Using String Kernels. Journal of Machine Learning Research 2, 419–444 (2002)
13. UCI Machine Learning Repository, `http://www.ics.uci.edu/~mlearn/MLRepository.html`
14. Kasabov, N.: Integrative Probabilistic Evolving Spiking Neural Networks Utilising Quantum Inspired Evolutionary Algorithm: A Computational Framework. In: Köppen, M., Kasabov, N., Coghill, G. (eds.) ICONIP 2008, Part I. LNCS, vol. 5506, pp. 3–13. Springer, Heidelberg (2009)